the text is written in sometimes very poor English --"It is exactly those processes which mediate (sic) proofs of theorems in mathematics that require (sic) that..."-- and their arguments are rambling. Supposing that they had something sensible to say we can only regret that they have buried it under so much insinuating verbiage. As it stands it leaves the reader wondering why they have put so much venom in their text, because they seem to have gone much farther than the usual practitioner's backlash.

None of the many papers about program verification and derivation that I have written or seen uses APL as a programming vehicle. This could be an accident, it could also be a consequence of the rich expression structure of APL. (They refer to proofs as "...substitutions to be checked with the aid of simple algebraic identities" which, in the case of APL-expressions are perhaps not so simple...) If the latter conjecture is correct, it would explain why APL-addicts might feel unhappy about (or threatened by) modern achievements in proving the correctness of (non-APL) programs. Does it help the understanding of this paper and its venom to know of the heavy involvement with APL among its authors? We can only guess and have our private opinions.

[1] DeMillo, Richard A., Lipton, Richard J., Perlis, Alan J., "Social Processes and Proofs of Theorems and Programs", Proceedings of the Fourth ACM Symposium on Principles of Programming Languages, pp. 206-214 (January 1977).

Plataanstraat 5
5671 AL NUENEN
The Netherlands

prof.dr.Edsger W.Dijkstra
Burroughs Research Fellow

****
* Response from R. A. DeMillo, *
* R. J. Lipton, and A. J. Perlis *
****

We are grateful to the editor of SEN for obtaining Professor Dijkstra's permission to publish [the above memorandum] and for giving us the opportunity to respond to it.

We must begin by refusing to concede that our confidence in a piece of real software has ever been increased by a proof of its correctness -- real software, such as sirline ticketing programs that issue real airline tickets or classroom scheduling programs that schedule real classes. We appreciate that Professor Dijkstra's memo was quite informal and was originally intended for those sympathetic to program verification, and among them it may be true that cases of raised confidence are well known and require no elaboration. But now that the correspondence has reached a larger audience, we hope that Professor Dijkstra will be quick to substantiate his assertion that such cases exist by pointing to some real-life examples.

In the form in which it is now expressed, the question about confidence may lead to an impasse: "Doesn't increase our confidence." "Does mine." "Doesn't." "Does." Our best recourse may be to the general practice of our peers. Do they act as if program verifications increased their confidence in programs? Do adherents of program verification verify their own programs? Our impression is that they do not, but we would be interested to learn otherwise. Or perhaps we should look to the lessons of history. Ralph London

[1] makes the interesting observation that "even as recently as the mid-1960's very few people knew what verification meant." He goes on to express surprise that "almost no program verifications were even attempted, and in fact the absence of verifications was not even noticed." We ourselves were not so surprised.

Professor Dijkstra says that "how to prove" the correct functioning of a piece of software is the subject of lively interchange among scientists in the field. True; we never said it wasn't. We said that the verifications themselves are long, ugly, and boring, no matter how concise, elegant, and fascinating the idea of verification may be. If verifications of real programs are currently being socialized, Professor Dijkstra should have no trouble pointing to the channels of communication; we look forward to his reply.

Somehow our distinction between "formal" and "understandable" has become confused in Professor Dijkstra's mind; since we are, to our regret, guilty of writing in sometimes very poor English, the fault may well be ours. Be that as it may, we do not believe that there is any antagonism between formalism and understandability. What we do say is that the two are not at all related. In the history of mathematics, formalism has served a single purpose: to freeze a concept, either because it had acquired so much stability that its presentation was no longer an issue, or because it needed to be made immutable so that it could be studied in the metamathematical sense. But it is abundantly clear that the axioms of group theory contribute very little to our understanding of groups; such understanding comes from studying invertible matrices, or the symmetries of geometric objects, or any of a dozen other sources. By the same token, knowing what constitutes a valid proof in first-order predicate calculus is very little help either in constructing or in understanding a real mathematical proof.

The most curious aspect of Professor Dijkstra's memo is his characterization of us as APL addicts. Since he brought it up, perhaps we should say that he is mistaken. We three might be able to mount a lively debate among ourselves on the virtues and vices of APL; such a debate would not, however, have even the slightest bearing on the "Social Processes" paper.

Ideally, any scientific statement should be open to critical review; ideally, the validity of any scientific statement should be independent of the personal characteristics and motives of its author. We all hold these ideals, but we all fail now and then to apply them to our own work. No matter how reasonable and impersonal and scientific an attack on our own ideas may be, it feels personal. It stings. Not feeling hurt is difficult, even if the attack is on the most trivial theorem or the most insignificant proof. No wonder that a criticism of so thoroughgoing and pervasive a worldview as that of program verification must seem to its adherents very ugly.

[1] R. London, Program Verification, in Research Directions in Software Technology (P. Wegner, ed.), MIT Press (1978).

Georgia Institute of Technology — Richard A. DeMillo

Yale University — Richard J. Lipton and Alan J. Perlis